Spring 2021

CSE 190

VR Technologies

Discussion 4

Guowei Yang
UCSDCSE

ANNOUNCEMENTS

- Homework 2 Released
  - Due **Sunday (5/2)**
  - VR Headset Required
  - Expect some minor updates on the specs page
- SAVE OFTEN

UCSDCSE

AGENDA

- Setting up Unity & Oculus Quest 2 for VR Support
- Build Virtual 3D Scene

UCSDCSE

SETTING UP UNITY & OCULUS QUEST

Setting Up Unity & Oculus Quest

- A little bit laborious, make sure follow all steps
- Ingredients
  - System: macOS/Windows 10/Linux
  - Unity ver. 2020.3.5f1
  - VR Headset (this discussion is mainly focused on Oculus Quest 2 development)
  - Cross-Platform: https://developer.oculus.com/documentation/unity/unity-cross-platform-dev/

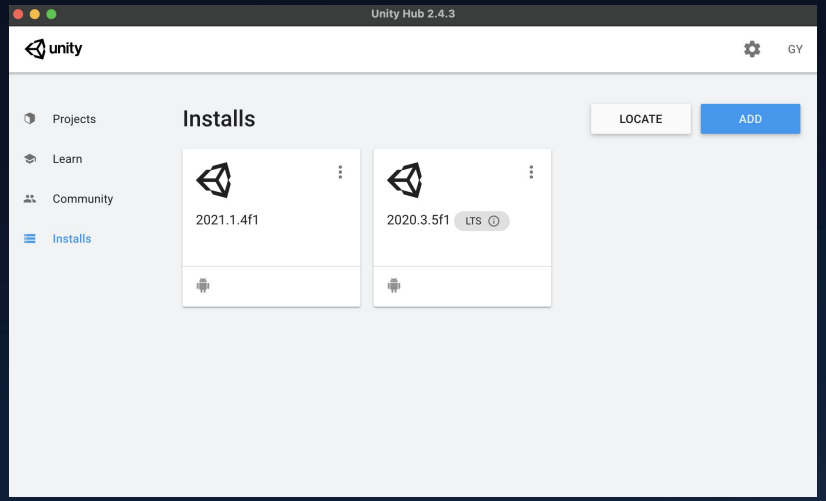
unity

+

oculus

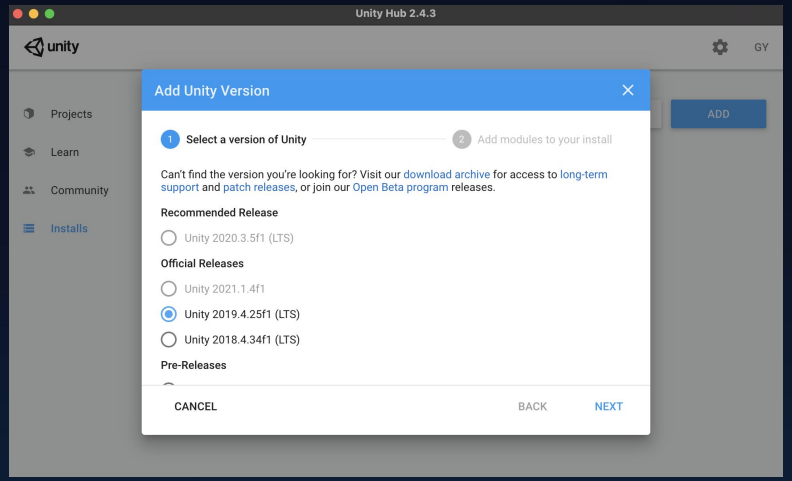
Setting Up - Install SDKs

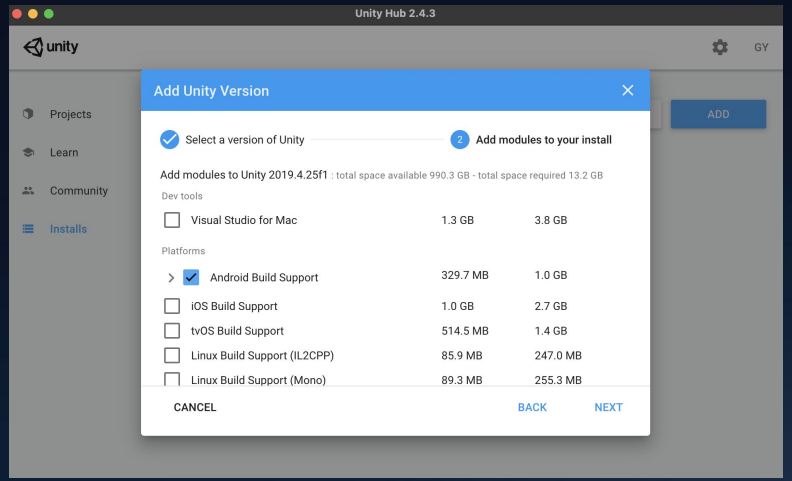
- Install Latest Version of Unity, Android SDK and JDK via Unity Hub
- Oculus Quest 2's OS is Android-Based



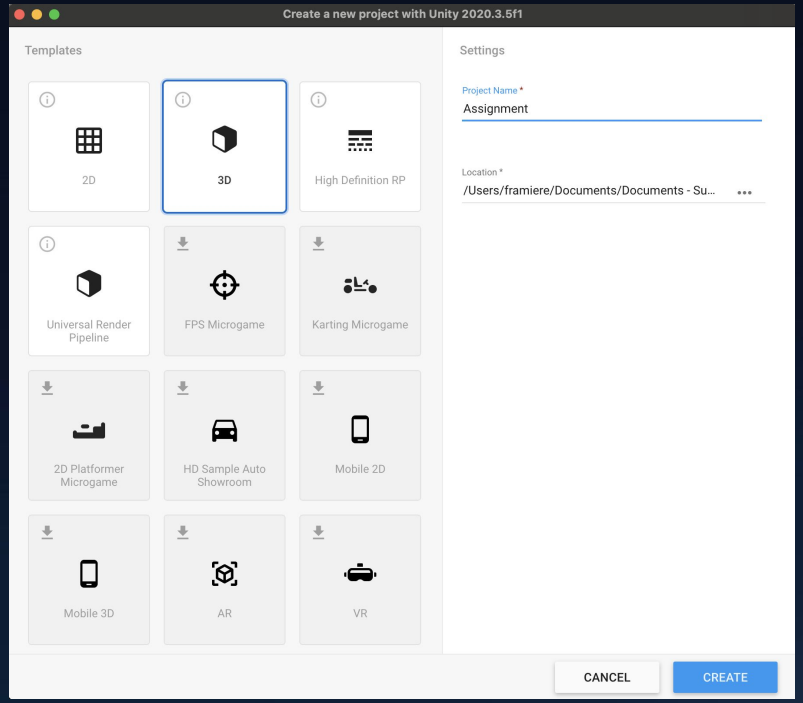
Unity Hub Install New Components

Setting Up - Install SDKs

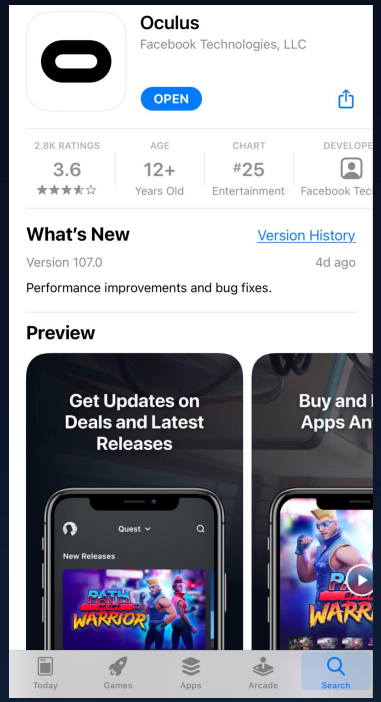


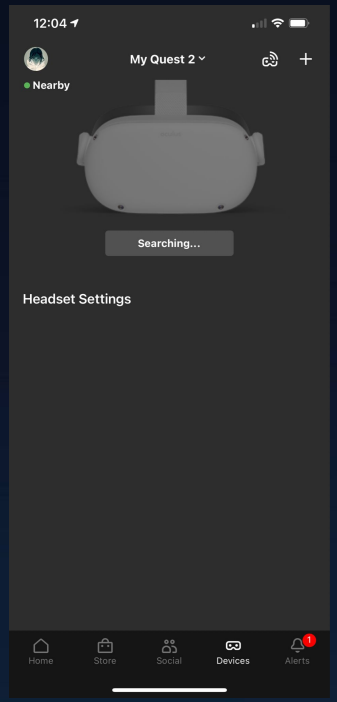


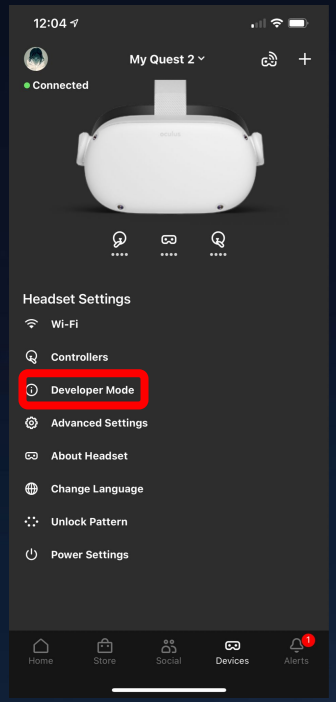
Setting Up - Create Unity Project for VR

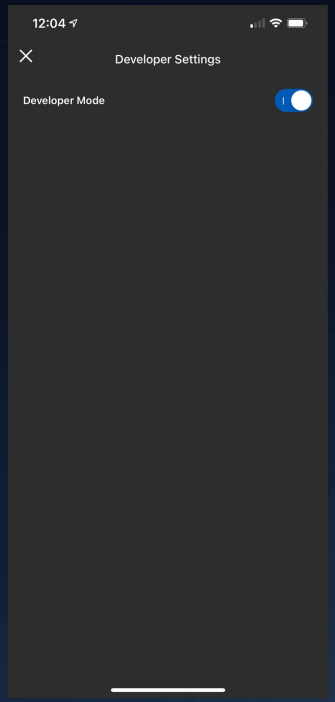


Setting Up - Enable Developer Mode for VR







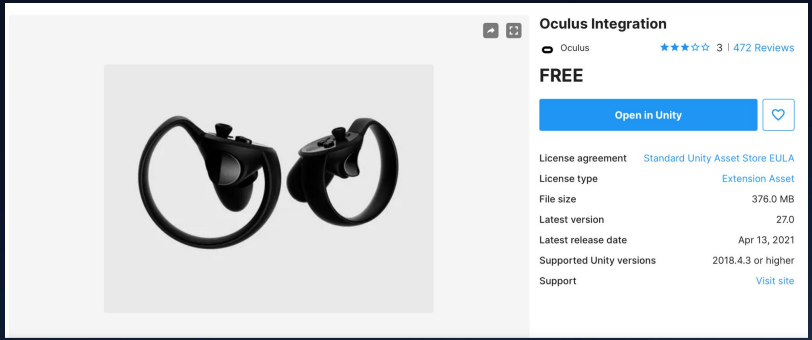


Setting Up - Unity Install Oculus Integration

- Oculus Integration: SDK for Unity provides support to develop Oculus apps in Unity
  - Handles headset tracking, controller tracking and so much more
- Installed from Unity Assets Store:

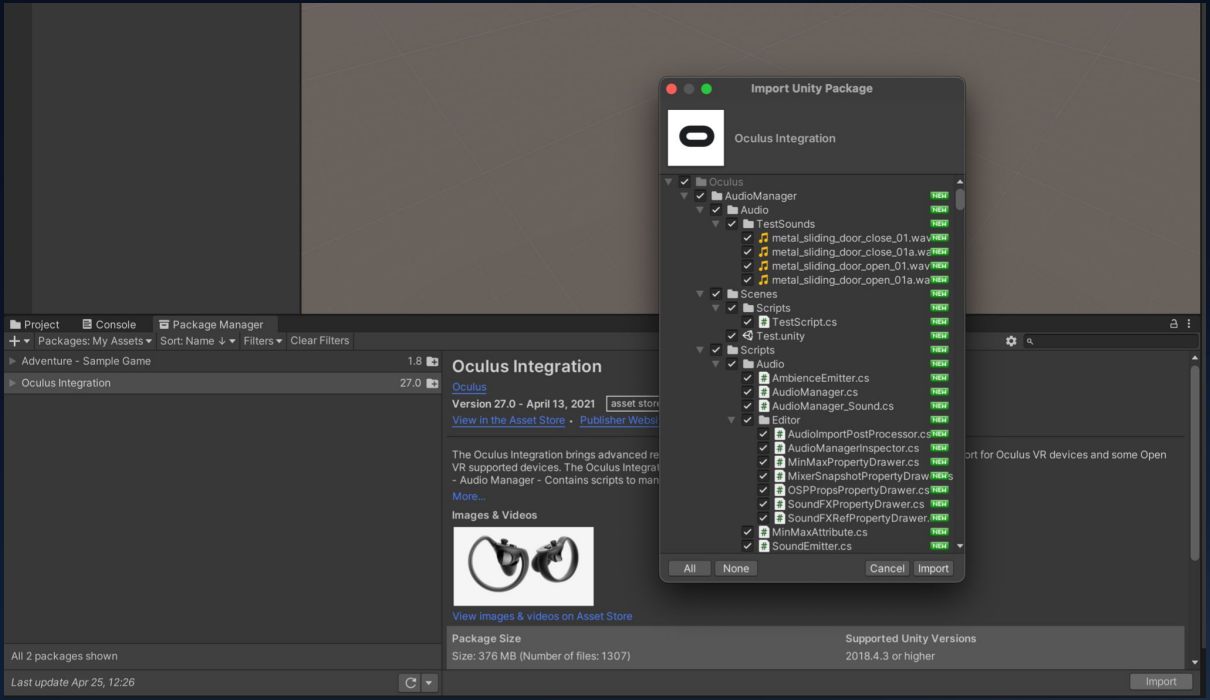
  https://assetstore.unity.com

  Version 27
- Import to your newly created project
- SAVE OFTEN



https://developer.oculus.com/downloads/package/unity-integration/

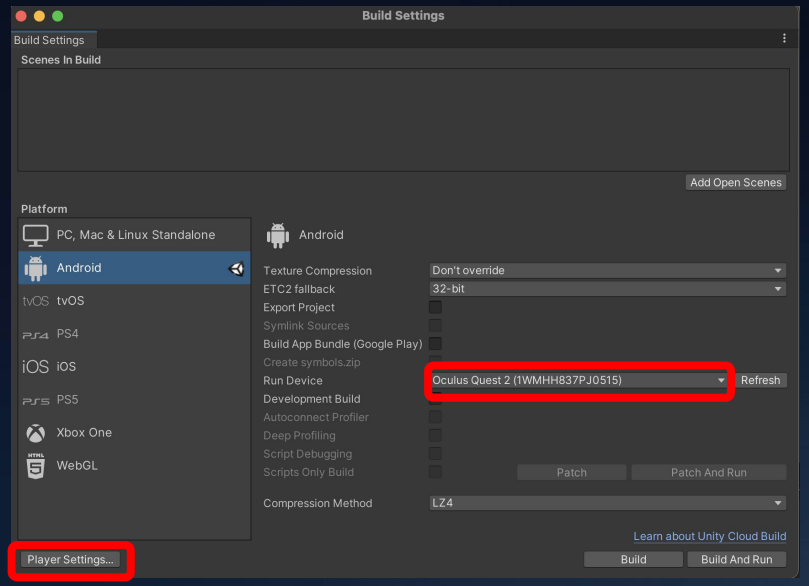
Setting Up - Unity Install Oculus Integration



Select all the components and import to your project

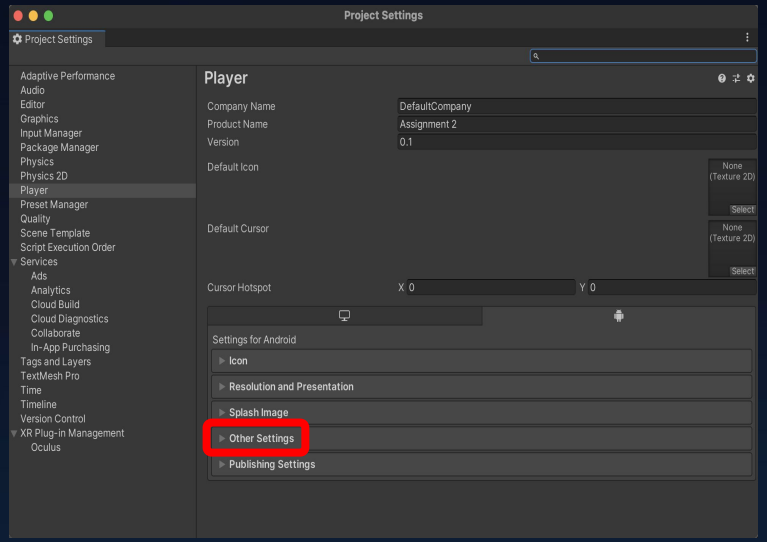
Setting Up - Configure Build Settings

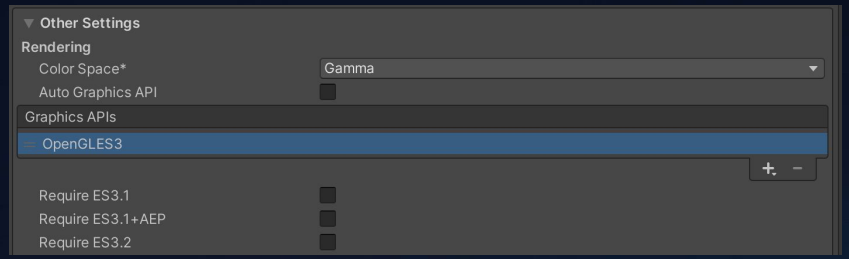
- File -> Build Settings -> Run Device
- Select your Oculus Quest (or other VR)
  - Make sure you installed Android SDK and JDK prior this step, otherwise Unity will complain it cannot find the SDKs
- SAVE OFTEN



Setting Up - Configure Build Settings

- Select the correct graphics API

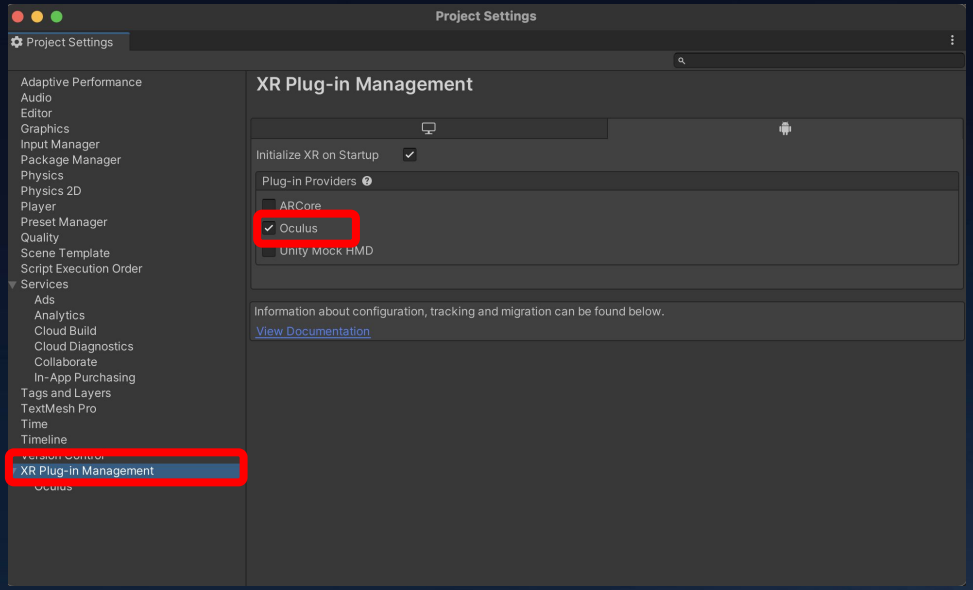




By default, the options here are Vulkan and OpenGL, make sure you drag OpenGL to the top (if Vulkan is present)

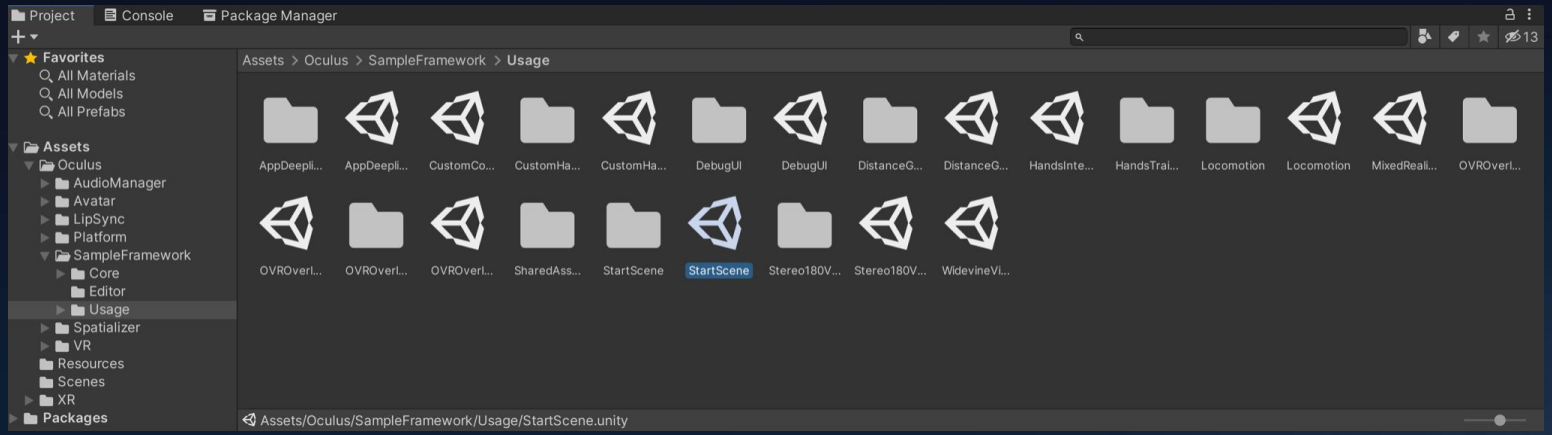
Setting Up - Configure Build Settings

- Enable Oculus Plug-in
- Black screen if missed this step



Setting Up - Test

- All those scenes should be able to be compiled and run on your headset
- If you encounter black screen issue, make sure to check XR Plugin Management is checked for Oculus
- SAVE OFTEN

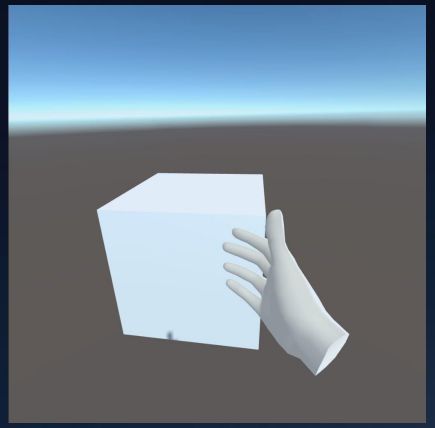


Setup Complete!

BUILD VIRTUAL 3D SCENE

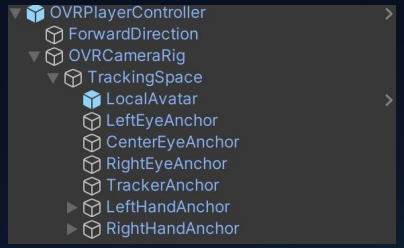
Build Virtual 3D Scene

- The Oculus Integration package adds scripts, prefabs, samples, and other resources to supplement Unity's built-in support.
- The package includes an interface for controlling VR camera behavior, a first-person control prefab, a unified input API for controllers, rendering features, debugging tools, and more
- SAVE OFTEN



Oculus Integration (OI) Components

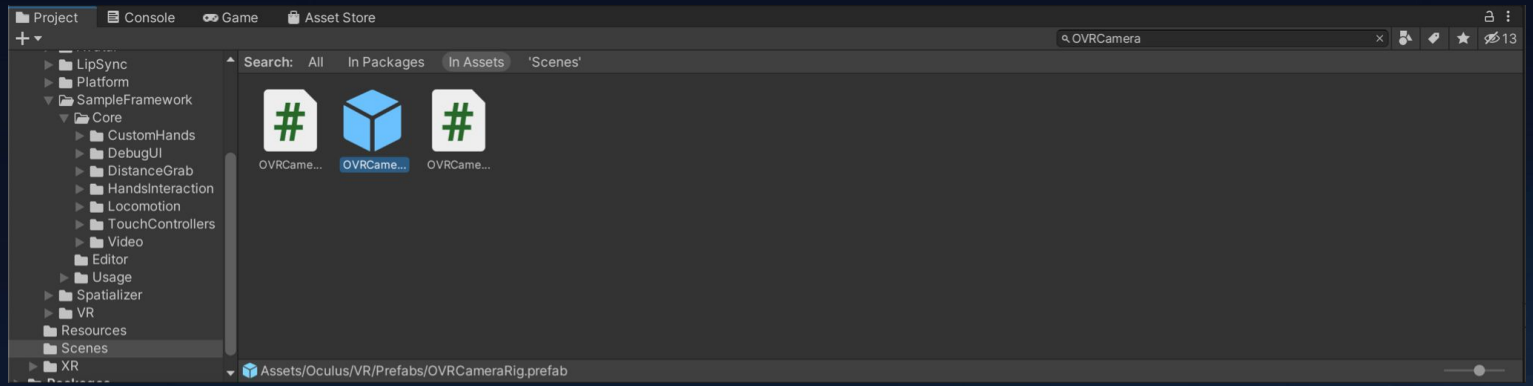
- OVRCameraRig
  - Prefab, provides the transform object to represent the Oculus tracking space.
  - Obtain headset physical poses, apply the value to the virtual camera
- OVRPlayerController
  - Contains a camera rig and a local avatar
- OVRHandPrefab
  - Enables hand tracking in Oculus Quest, can control user interface with bare hands
- LocalAvatar
  - Prefab, “you” in the virtual world, can see your own hands in game



Select necessary components and add it to the scene hierarchy

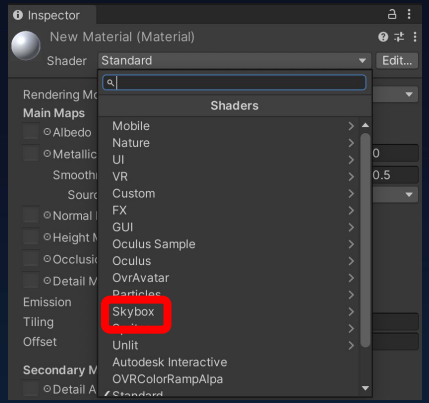
Oculus Integration (OI) Components

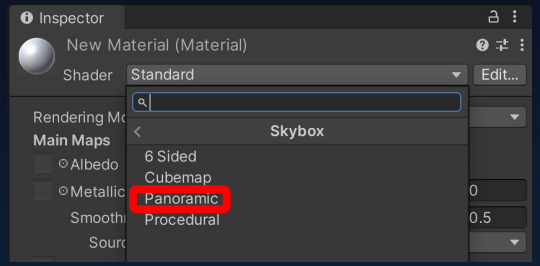
- Use the search bar to find necessary components needed for the project



Build Virtual 3D Scene - Skybox

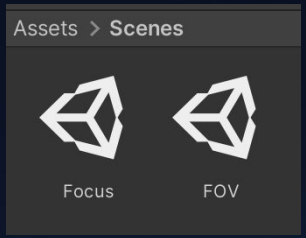
- A Skybox is a 6-sided cube that is drawn behind all graphics in the game.
- Used as 360 degree background
- Create new Material Asset -> Skybox

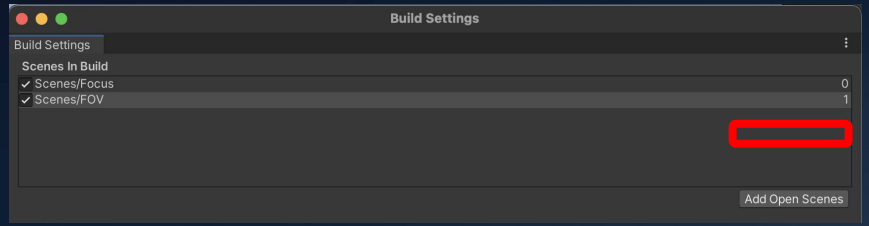




Build Virtual 3D Scene - Scene Switch

- Unity provides mechanisms for switching scenes in script
- Suitable for this project as you are asked to create several virtual scenes for measurement
- Procedure:
  - Have multiple scenes readily available
  - Make a GameObject to be the trigger for the scene switch
  - Attach a C# script to the object
  - `using UnityEngine.SceneManagement`
  - `SceneManager.LoadScene(sceneName);`
- Add all the scenes to the build phase





Useful APIs

: API Package that provides raw input from the tracked controllers

Controller Tracking: `OVRInput` https://developer.oculus.com/reference/unity/v27/class_o_v_r_input

Following APIs provides the position and rotation of the controllers in tracking space

- `OVRInput.GetLocalControllerPosition( OVRInput.Controller controllerType )`
- `GetLocalControllerRotation ( OVRInput.Controller controllerType )`

Headset Tracking: `OVRCameraRig` https://developer.oculus.com/reference/unity/v27/class_o_v_r_camera_rig

- ```
  OVRCameraRig overCameraRig;  //Obtain this using GetComponent
  var position = overCameraRig.centerEyeAnchor.position;
  ```
- `centerEye`, `leftEye`, `rightEye` for different purposes
  ```

GOOD LUCK!

QUESTIONS?